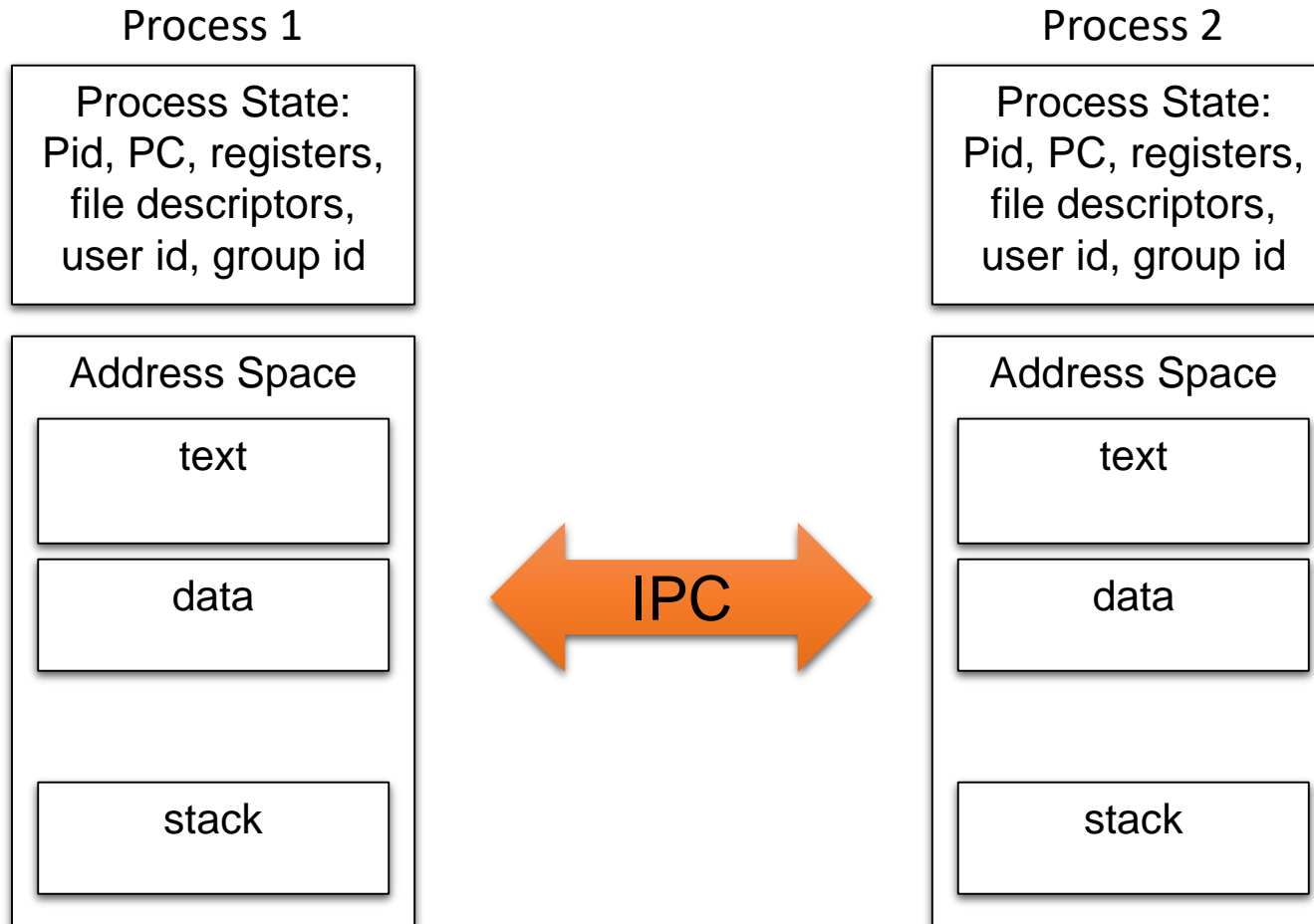
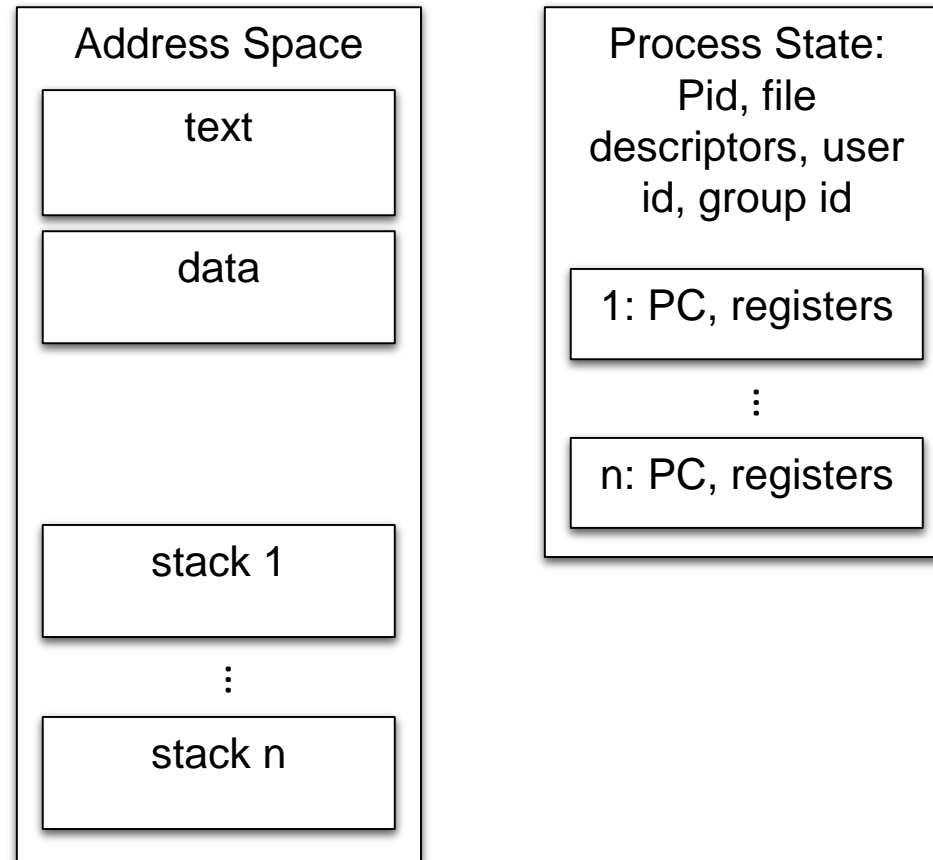


# Distintos procesos



# Multithreading



# Threads



- Cada thread mantiene sus propios:
  - Stack Pointer
  - Registers, PC
  - Scheduling properties
  - Signals behavior
  - Thread specific data
- Comparte:
  - Address Space
  - File descriptors
  - Process properties (pid, uid, gid, etc.)

# POSIX Threads

- Creación de un thread

```
#include <pthread.h>
```

```
int pthread_create(pthread_t *thread,  
                   const pthread_attr_t *attr,  
                   void *(*start_routine) (void *),  
                   void *arg);
```



- Linkar con -lpthread 


# POSIX Threads

- Terminación de un thread
  - Llamado explícito a:

```
void pthread_exit(void *retval);
```

- Retorno de `start_routine`.
  - Cancelada con `pthread_cancel`.
  - Terminación del proceso.
- `pthread_self()` retorna el pthread ID del calling thread.
- `pthread_equal()` compara pthread IDs de manera portable.

# POSIX Threads

- Cuando un thread termina, el comportamiento es similar a los procesos zombies. 
- `pthread_join` es similar a `wait` de un proceso hijo muerto. Espera por un thread terminado.
- Detached threads: liberación automática de recursos.
- No esperan por otro thread que las joinee.
- `pthread_detach`
- Una vez detachado un thread, no se puede joinear.

# POSIX Threads

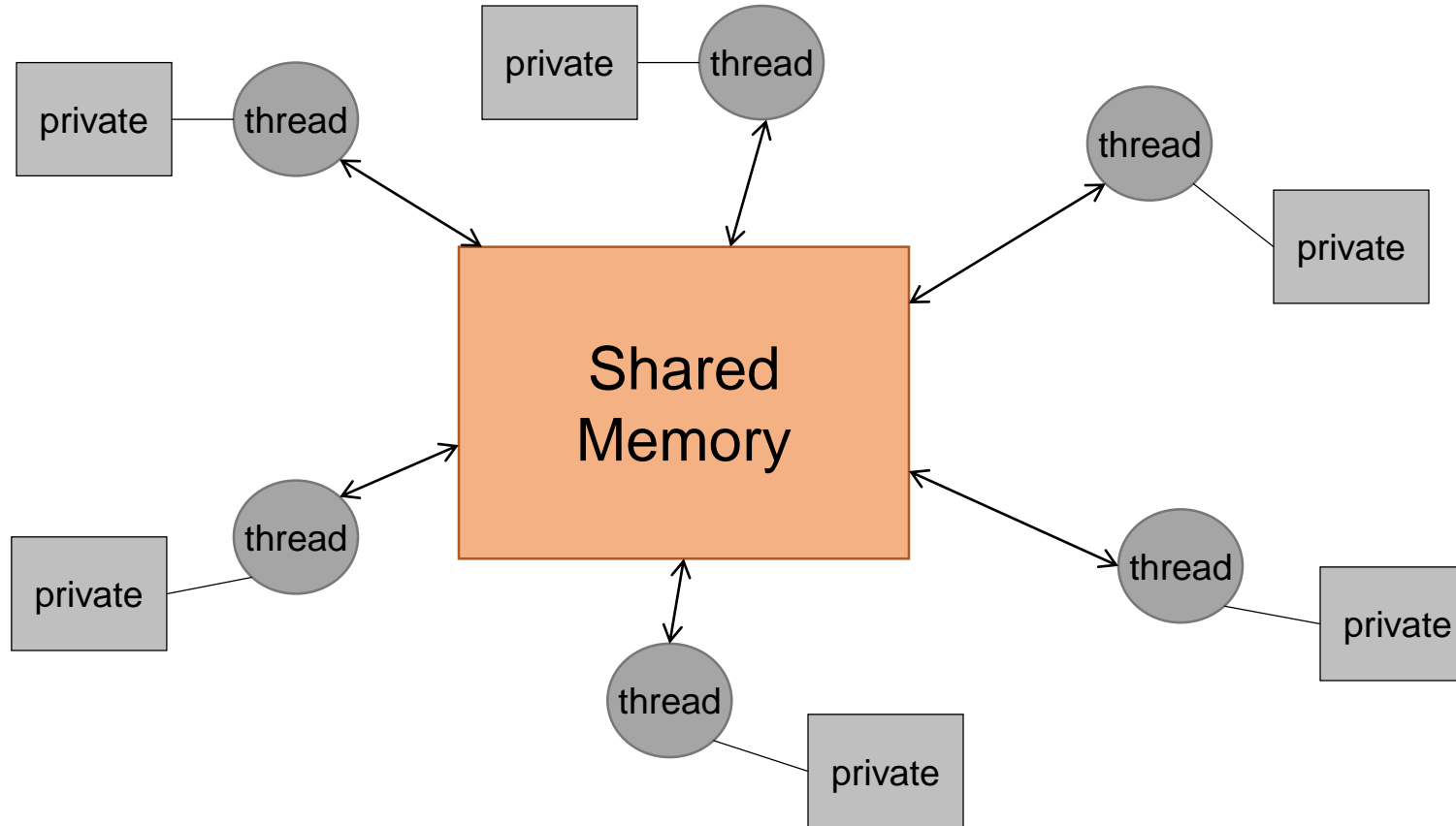
- Thread attributes: `pthread_attr_t`
- Inicializados con `pthread_attr_init`
- Destruído con `pthread_attr_destroy`
- Atributos posibles:
  - `pthread_attr_setdetachstate`
  - `pthread_attr_setguardsize`
  - `pthread_attr_setinheritsched`
  - `pthread_attr_setschedparam`
  - `pthread_attr_setschedpolicy`
  - `pthread_attr_setscope`
  - `pthread_attr_setstack`
  - `pthread_attr_setstacksize`

# Threads

- Permiten paralelizar la programación. Modelos:
  - Manager/workers
  - Pipeline
- Ventajas en manejo de:
  - Eventos asincrónicos
  - Bloqueos por I/O.
  - Prioritización
  - Paralelismo en ejecución u operaciones sobre datos.



# Threads – Shared Memory Model



# Threads - Problemas

- Reentrancy: las funciones deben poder ser ejecutadas desde distintos threads concurrentemente.
  - No guardar estado estático entre llamados.
  - No retornar nada asociado a buffers propios.
    - `man -k _r`
- Thread safeness: protección de datos compartidos.
- Race conditions.
- Mutual exclusion. Serialización de acceso.



# Mutex

- Recurso que permite sincronizar threads.
- Un solo thread puede tomar el mutex al mismo tiempo.
- Operaciones: lock y unlock.
- Similar a un semáforo con cuenta máxima de 1.
- API:

```
pthread_mutex_init  
pthread_mutex_destroy
```

```
pthread_mutex_lock  
pthread_mutex_trylock  
pthread_mutex_timedlock
```

```
pthread_mutex_unlock
```

# Condition Variables

- Recurso que permite sincronizar threads basándose en eventos. Un thread espera hasta que una condición sea cierta. De otra manera debería hacer polling.
- Las variables de condición están asociadas a un mutex.
- API:

```
pthread_cond_init  
pthread_cond_destroy
```

```
pthread_cond_wait  
pthread_cond_timedwait  
pthread_cond_signal  
pthread_cond_broadcast
```

# Barriers

- Recurso que permite sincronizar threads basándose en que  $n$  threads arriben al punto de sincronización.
- API:

```
pthread_barrier_init  
pthread_barrier_destroy
```



```
pthread_barrier_wait
```

# Read/Write Locks

- Recurso que permite sincronizar threads permitiendo múltiples accesos de threads para lectura (compartido) y único para escritura (exclusivo).
- API:

```
pthread_rwlock_init  
pthread_rwlock_destroy
```

```
pthread_rwlock_rdlock  
pthread_rwlock_wrlock  
pthread_rwlock_unlock
```

```
pthread_rwlock_tryrdlock    pthread_rwlock_trywrlock  
pthread_rwlock_timedrdlock  pthread_rwlock_timedwrlock
```

# Spinlocks

- Recurso que permite sincronizar threads.
- La espera por el recurso se hace “spinning” (busy waiting). Evita context switching.
- Tiene sentido en multiprocesadores.
- API:

```
pthread_spin_init  
pthread_spin_destroy
```

```
pthread_spin_lock  
pthread_spin_trylock  
pthread_spin_unlock
```